Atty. Docket No.: 4860[redacted]937

Patent

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

BOARD OF PATENT APPEALS AND INTERFERENCES

In Re Application of:

Guy Riddle

Serial No.: 08/646,503

Filed: May 8, 1996

For: **METHOD AND APPARATUS FOR DYNAMIC LAUNCHING OF A TELECONFERENCING APPLICATION UPON RECEIPT OF A CALL**

Examiner: Ho, C.

Art Unit: 2757

APPEAL BRIEF
IN SUPPORT OF APPELLANT'S APPEAL
TO THE BOARD OF PATENT APPEALS AND INTERFERENCES

Assistant Commissioner of Patents
Washington, D.C. 20231

Dear Sir:

The Appellant hereby submits this Brief in triplicate in support of their appeal from a final decision by the Examiner, mailed on February 17, 2000, which was reiterated in an Advisory Action, mailed on May 8, 2000, in the above-captioned case. The Appellant respectfully requests consideration of this appeal by the Board of Patent Appeals and Interferences for allowance of the above-captioned patent application.

FIRST CLASS CERTIFICATE OF MAILING

I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail with sufficient postage in an envelope addressed to the Assistant Commissioner for Patents, Washington, D.C. 20231

on August 15, 2000
Date of Deposit

Carrie Boccaccini
Name of Person Mailing Correspondence

Signature

8-15-2000
Date

TABLE OF CONTENTS

I. Real Party in Interest

The real party in interest is Apple Computer, Inc. of Cupertino, California.

II. Related Appeals and Interferences

There are no related appeals or interferences.

III. Status of Claims

Claims 1-23 stand rejected.

IV. Status of Amendments

No amendments have been filed subsequent to the Final Rejection mailed on February 17, 2000. A copy of all claims on appeal is included in Appendix A.

V. Summary of the Invention

The invention defined in the appealed claims concerns methods and apparatuses for dynamically launching teleconferencing applications upon receipt of a call. A system in accordance with the present invention includes a call director, a demon conference component (i.e., a conference component acting in demon mode), a transport component and a network component (p. 13, lines 6-9). The call director performs several functions, such as initiating the automatic launching of a conferencing application when a call is received by the system and initiating and interacting with the demon conference component to control the transfer of calls to a conferencing application (p. 13, lines 16-21). The demon conference component engages in "persistent listening" of incoming calls (p. 14, lines 5-8). The demon conference component communicates with other conference components to transfer incoming calls

indicated by the transport component and the network component using a shared data structure in memory (p. 14, lines 21-24).

When a conferencing application requests persistent listening, the demon conference component performs persistent listening on behalf of the conferencing application which requested it. The demon conference component will perform persistent listening on the port specified in the listen string sent from the conferencing application. The request includes both an application signature, which identifies the conferencing application as the requester, and the listen string. A separate conference component passes the persistent listening request from the conferencing application to the demon conference component. (p. 17, lines 5-19). The demon conferencing component notifies the call director that a conferencing application has requested persistent listening and sends the application signature and listen string (p. 17, lines 22-27). After sending the request for persistent listening, the conferencing application may end execution or continue to run (p. 18, lines 10-11).

When an incoming call is detected by the transport component, the demon conference component transfers the incoming call to the separate conference component which notifies the conferencing application (assuming it is still running) of the incoming call (p. 21, lines 3-8). The incoming call is transferred to the conferencing application from the transport component by a command from the conference component (p. 21, lines 19-22). If the conferencing application is not running when the incoming call is detected, the call director will attempt to locate and launch the conferencing application (p. 26, lines 22-27). The call director determines if the conferencing application is listening in the same way as it was when it requested persistent listening. If the conferencing application is listening in the same way, then the incoming call is transferred to the conferencing application. (p. 28, lines 12-26).

VI. Issues

A. Whether claims 1-23 are patentable under 35 U.S.C. § 103(a) over Mirashrafi et al., U.S. Patent No. 5,574,934 (Mirashrafi), in view of Andersen et al., U.S. Patent No. 5,674,003 (Andersen), and Jamsa et al., "Internet Programming" (Jamsa).

B. Whether claims 7, 16 and 23 are patentable under 35 U.S.C. § 103(a) over Mirashrafi alone.

VII. Grouping of Claims

For the purpose of this appeal, claims 1-23 stand or fall together.

VIII. Argument

A. Rejection of Claims 1-23 Under 35 U.S.C. § 103(a) in View of Mirashrafi, Andersen And Jamsa Is Improper Because It Would Not Have Been Obvious to Achieve the Features of Claims 1-23 in View of Mirashrafi, Andersen And Jamsa

Mirashrafi is directed to preemptive priority-based signal transmission. Specifically, each type of signal is assigned a priority level and signals of a certain type are transmitted as they become ready for transmission. However, if signals of a different type having a greater priority become ready for transmission, then the transmission of the lower priority signals is interrupted to allow transmission of the higher priority signals. The transmission of the lower priority signals resumes after the completion of the transmission of the higher priority signals.

Andersen is directed to communicating data streams from at least a first computer process operating on a computer using a socket based transport interface to at least a second computer process operating on another computer using a socket

based transport interface. Socket groups are created for use by the at least first and second computer processes. A minimum quality-of-service is associated with one of the socket groups, and a telephony connection meeting the minimum quality-of-service is established between the computers. Each socket group includes a plurality of socket communication endpoints which are connected to one another by a plurality of communication channels. Thus, one computer may communicate with another computer. Accordingly, a plurality of data streams may be transmitted between the at least first and second computer processes via the communication channels over the telephony connection.

Jamsa provides an overview of internet programming. Specifically, Jamsa explains various aspects of sockets, such as the theory behind sockets and the proper use of sockets. Topics such as configuring sockets, connecting sockets, transmitting data through sockets, and receiving data through sockets are discussed in detail. Sockets are also explained in conjunction with Microsoft Windows protocols.

1. Mirashrafi, Andersen and Jamsa Fail to Disclose, Teach or Suggest, Either Individually or in Combination, A Listen String As Claimed in the Present Invention

Appellant's independent claims 1, 8-10 and 17 claim a "listen string containing an application signature, an application signal type and an application signal port." The listen string includes such information to direct the persistent listening performed by the demon conference component and to facilitate the proper transfer of an incoming call.

Appellant respectfully submits that none of Mirashrafi, Andersen and Jamsa discloses, teaches or suggests Appellant's claimed listen string.

Regarding Mirashrafi, the Examiner has pointed to an "associated usage ID" to provide support for the assertion that Mirashrafi discloses the establishment of a listen string as claimed by Appellant (see 5/8/00 Advisory Action, paper no. 16, p. 2). Furthermore, the Examiner has cited numerous passages in Mirashrafi in support of such assertion. However, none of the cited passages, which are listed below, provides any indication that the associated usage ID of Mirashrafi is similar to Appellant's claimed listen string.

- column 19, lines 22-25:

> In general, conferencing system 100 is capable of simultaneously supporting multiple applications that support different types of conferencing services (e.g., audio/video conferencing, data sharing, and background file transfer).

This passage simply provides an overview of conferencing system 100 without indicating that conferencing system 100 features a listen string containing an application signature, an application signal type and an application signal port.

- column 22, lines 23-25:

> Each channel pair has an associated usage ID which is defined by the application.

The associated usage ID disclosed in Mirashrafi allows an application to define a channel pair that is established by the application such that other applications will know the purpose of the channel pair. However, there is no indication that the associated usage ID is included in a listen string. Even if the associated usage ID is considered to be similar to Appellant's claimed application signature, Mirashrafi does not disclose, teach or suggest using the associated usage ID as part of a listen string.

- column 23, lines 20-25:

> Which application is establishing a channel pair (which is important for CMDLL on the remote site so that it knows which application to notify).
> The usage id for the channel pair (which is important for the remote application so that it knows what to do with the channel pair).

This passage describes what is identified by the ID field. According to the cited passage, the ID field identifies the application associated with a channel pair and the usage of the channel pair. However, there is simply no indication that the ID field or the associated usage ID is included as part of a listen string.

- columns 25-26

Columns 25 and 26 describe "On Demand Application Invocation" and "Managing Multiple Applications." The on demand application invocation applies when a conferencing application running in a first conferencing system attempts to establish contact with a second conferencing system where the conferencing application is not currently running. The conferencing application on the first conferencing system sends a communication request to the second conferencing system. The application ID of the conferencing application is received by the second conferencing system along with the communication request. However, there is no indication that the second conferencing system listens for the communication request using any feature similar to Appellant's claimed listen string which includes an application signature, an application signal type and an application signal port.

- column 27, lines 18-27:

> Data conferencing application 504 knows that the connection has been established, because application 504 has already registered with the conference manager 544 and the conference manager 544 informs all registered applications of connections by sending the CMN_CALL message. Since data conferencing application 504 already knows that the connection has been established, application 504 makes the channel request by calling the cmGetChannelPair function of the conference manager 544.

This passage describes the operations performed by data conferencing application 504 once a connection has been established. The conference manager 544 informs registered applications, such as application 504, of connections once they are established. Once application 504 is so informed, application 504 makes a channel request in preparation of communicating with another application. There is no indication that application 504 listens for calls or updates from conference manager 544 using a listen string as claimed by Appellant.

- column 70, lines 61-65:

> The logical channels provide the primary mechanism clients use to send multiple data types (e.g., audio, video, data). The layer services multiplex these data types together for transmission to the remote sites.

This passage simply describes the function of the channels which are established between conferencing systems at different locations. Again, there is no indication of a listen string as claimed by Appellant. Appellant respectfully submits that sending different data types across channels does not suggest the use of a listen string containing an application signature, an application signal type and an application signal port.

- column 84, lines 63-66:

> When a conferencing application is installed, the above information is added to the CM.INI section. An application ID is assigned to all conferencing applications that want to work with the conference manager 544.

This passage describes how conferencing application information is added to the initialization file associated with the conference manager 544. Although conference manager 544 has access to information about the conferencing applications, there is no indication that conference manager 544 uses a listen string as claimed by appellant. According to what is disclosed in other portions of Mirashrafi (for example, see col. 27, lines 18-27), it is apparent that conference manager 544 uses the stored conferencing application information to notify applications of established connections. If a conferencing application's information is not stored in CM.INI, then the conferencing application may not be recognized by conference manager 544 (see col. 87, lines 56-57). However, there is no indication that the CM.INI file is a listen string as claimed by Appellant or that conference manager 544 uses the CM.INI as a listen string as claimed by Appellant.

- column 86, lines 2-4:

> The Conference Manager combines the application ID provided in cmRegister to form a channel ID.

Appellant respectfully submits that using an application ID to form a channel ID as disclosed in Mirashrafi fails to suggest using a listen string containing an application signature, an application signal type and an application

signal port. There has been no showing that the application ID of Mirashrafi is used as part of a listen string.

- column 87, lines 55-65:

> A conferencing application will not be recognized by the conference manager 544 unless it is properly installed. If an application with the specified ID is already installed, then the current information in the .INI file is overwritten by the new information. Application IDs range from 0 through 1023. See also cmRegister. The cmInstall function determines if the application has already been installed in the global LIB data structure by comparing the application ID. If so, the function updates with the application name and path. If not, the function adds the application to the global LIB data structure.

This passage describes the installation of and subsequent identification of a conferencing application. Proper installation of a conferencing application results in the entering of information into the CM.INI file and the global LIB data structure. Once a conferencing application's application ID is entered into the CM.INI file, the conferencing application is recognized by the conference manager 544. Reinstallation of a conferencing application updates the information associated with the conferencing application. There is no indication that the CM.INI file is a listen string which contains application IDs, or that application IDs are used as part of a listen string, or that conference manager 544 uses a listen string, or that the global LIB data structure is used as part of a listen string. Appellant respectfully submits that identifying a conferencing application as disclosed in Mirashrafi fails to suggest using a listen string containing an application signature, an application signal type and an application signal port.

- column 88, lines 19-20:

> The cmRegister function is used to register a conferencing application and is defined as follows:

Conferencing applications call the cmRegister function when they are invoked (see col. 88, lines 40-45). If a conferencing application has been installed and its corresponding application ID is found in the CM.INI file, then the conferencing application is registered with the conference manager. If the conferencing application has not been installed or its corresponding application ID is not found in the CM.INI file, then a CMRC_UNKNOWNAPP value is returned. (see col. 88, lines 50-66). Although Mirashrafi repeatedly discloses the use of application IDs with regard to installing, registering and notifying conferencing applications, Mirashrafi also repeatedly fails to disclose, teach or suggest the use of its application IDs as part of a listen string as claimed by Appellant. As such, Mirashrafi fails to disclose, teach or suggest expressly recited elements of Appellant's claimed invention.

Although the above cited passages represent only a portion of Mirashrafi, the other portions of Mirashrafi are similarly deficient. Furthermore, even when the above cited passages are read in view of one another, there is still no suggestion to use a listen string containing an application signature, an application signal type and an application signal port.

Regarding Andersen, the Examiner has cited numerous passages to support the assertion that Andersen discloses a listen string containing an application type and an application signal port (see 5/8/00 Advisory Action, paper no. 16, p. 2). However, none of the cited passages, which are listed below, provides any indication that Andersen discloses, teaches or suggests Appellant's claimed listen string.

- column 15, lines 20-27:

> FIG. 4 shows a flowchart 400 of the procedural steps which may be undertaken during a conferencing session utilizing application software employing an embodiment of the present invention. A video conferencing application is launched at step 402, which creates an audio socket and video socket (step 404) for establishing communication with a remote endpoint (also running a conferencing application).

According to this passage, the launching of a video conferencing application creates audio and video sockets for communicating with another conferencing application. The flowchart 400 depicts a number of steps in a conferencing session, such as: requesting a connection to a remote endpoint; dialing the remote endpoint; establishing a telephony connection; and transmitting audio and video data according to socket priorities. However, there is no indication in the flowchart 400 or in the above cited passage that a listen string as claimed by Appellant is used.

- column 15, lines 35-38:

> The quality-of-service which is desired for the socket group is established by the conferencing application as a flow spec and is stored at a location indicated by the pointer CpA-Flowspec (see WSAConnectEx() function).

This passage simply describes the establishment of a quality-of-service level for a socket group. An established telephony connection must meet the minimum quality-of-service level that is specified for a socket group. There is no indication that a quality-of-service level has any bearing on a listen string as claimed by Appellant.

- column 3, lines 10-18:

> An example of where the method of the invention can be advantageously employed is a computer conferencing system, wherein video data, audio data and graphical and/or text data is transmitted via a telephony connection between two computers. In this instance a separate socket communication endpoint is created at each computer for each of the video, audio and graphics/text data streams, and connections established between the respective sockets by way of the telephony connection.

This passage describes generally how each type of data stream may have its own socket communication endpoint at each computer. According to Andersen, such a configuration may provide more efficient transfer of data. However, there is no indication that a socket or socket endpoint uses a listen string when listening for incoming connections.

- column 3, lines 19-24:

> The separate data streams are multiplexed for transmission over the telephony connection and, in normal operation, the bandwidth of the telephony connection is shared between all of the data streams from the sockets comprising the socket groups associated with the conferencing application.

This passage describes generally the transmission of separate data streams over a telephony connection. Again, there is no indication that a socket or socket endpoint uses a listen string when listening for incoming connections.

- column 7, lines 64-67:

> The remote site application then listens for incoming connections on the sockets (step 306) using the listen() function, and waits until a connection is detected (steps 308 and 310).

Although Andersen discloses that a remote site conferencing application listens for incoming connections on the created sockets, there is no indication that a conferencing application listens using a listen string as claimed by Appellant. The listen() function is not described in any further detail, and the bare recitation of a listen function does not suggest the use of a listen string as claimed by Appellant.

- column 8, lines 1-7:

> When the local site computer wishes to communicate with the remote site, it creates at least one socket (step 352) to serve as a local endpoint for communication with the remote site application. The local site computer, using the telephony service provider and, for example, functions provided by TAPI proceeds to establish a telephony connection to the remote site application.

This passage describes how a local site computer/application may communicate with a remote site computer/application. A local socket/endpoint is created to communicate with a remote socket/endpoint. A telephony connection bridges the local socket/endpoint to the remote socket/endpoint. However, there is no indication that a remote site computer/application listens for an incoming connection from a local site computer/application using a listen string as claimed by Appellant.

Although the above cited passages from Andersen represent only a portion of Andersen, the other portions of Andersen are similarly deficient. Furthermore, even when the above cited passages are read in view of one another, there is still no suggestion to use a listen string containing an application signature, an application signal type and an application signal port.

Regarding Jamsa, Appellant respectfully submits that Jamsa also fails to disclose, teach or suggest a listen string containing an application signature, an

application signal type and an application signal port. Jamsa describes various aspects of sockets, such as configuring a socket, connecting a socket and transmitting data through a socket. Jamsa also discloses the use of a listen function. For example, Jamsa states that a "connectionless client must listen at a protocol port for a reply datagram" (p. 166, lines 25-26). Jamsa further states that "the listen function tells the socket to 'listen' for incoming connections and to acknowledge connection requests" (p. 172, lines 2-3). However, Jamsa fails to indicate that the listen function operates according to a listen string containing each of an application signature, an application signal type and an application signal port. Jamsa only discloses that "[t]he listen function requires two parameters: socket handle and queue length" (p. 174, line 27). There is no suggestion of parameters which are similar to each of Appellant's claimed application signature, application signal type and application signal port, all of which are contained in a listen string. Accordingly, Jamsa fails to disclose, teach or suggest Appellant's claimed listen string.

As discussed above, each of Mirashrafi, Andersen and Jamsa fails to disclose, teach or suggest Appellant's claimed listen string. Accordingly, even if it would have been obvious to one of ordinary skill in the art at the time of Appellant's claimed invention to combine Mirashrafi, Andersen and Jamsa, a combination of such references still would not have yielded Appellant's claimed invention. Thus, the rejection of Appellant's claims 1-23 under 35 U.S.C. § 103(a) in view of Mirashrafi, Andersen and Jamsa is unsupportable, and claims 1-23 are allowable over Mirashrafi, Andersen and Jamsa.

B. Rejection of Claims 7, 16 and 23 Under 35 U.S.C. § 103(a) in View of Mirashrafi Alone Is Improper Because Mirashrafi Fails to Disclose, Teach or Suggest Expressly Recited Elements of Independent Claims 1, 10 and 17

Claims 7, 16 and 23 depend from independent claims 1, 10 and 17, respectively, and therefore incorporate all of the elements of their respective independent claims in addition to adding elements of their own. As previously stated, Mirashrafi fails to disclose, teach or suggest expressly recited elements of claims 1, 10 and 17. Specifically, Mirashrafi fails to disclose, teach or suggest a listen string containing an application signature, an application signal type and an application signal port. Because claims 7, 16 and 23 incorporate the elements of claims 1, 10 and 17, respectively, Mirashrafi also fails to disclose, teach or suggest Appellant's claimed listen string in claims 7, 16 and 23. As such, Appellant's claims 7, 16 and 23 are not rendered obvious by Mirashrafi alone and are therefore allowable over Mirashrafi.

IX. Conclusion

For the foregoing reasons, Appellant respectfully submits that claims 1-23 overcome Mirashrafi, Andersen and Jamsa and are therefore patentable. Any dependent claim not specifically addressed is deemed allowable in view of its dependency from an independent claim as argued above. For the reasons presented herein, the removal of the present rejections and allowance of the present claims is respectfully requested.

Charge Our Deposit Account

If there are any further charges not accounted for herein, please charge them to our deposit account No. 02-2666.

Respectfully submitted,

BLAKELY, SOKOLOFF, TAYLOR, & ZAFMAN LLP

Dated: 8/15/00

By: _____

Andrew C. Chen
Reg. No. 43,544

X. Appendix A

The claims on appeal read as follows:

1. In a computer system having a memory, a processor, and a network interface, a method comprising:

launching a call director unit to set up a demon conference component in said memory;

receiving an incoming call signal on said network interface;

processing said incoming call signal in said demon conference component to detect an intended recipient application using a listen string, said listen string containing an application signature, an application signal type and an application signal port; and

launching said intended recipient application using said application signature.

2. The method of claim 1, wherein said processing said incoming call signal comprises:

parsing said incoming call signal to determine a signal type and a signal port; and

determining said intended recipient application based on said signal type and said signal port.

3. The method of claim 1, wherein said launching said intended recipient application comprises:

determining said intended recipient application based on a signal type and a signal port;

locating said intended recipient application using said application signature; and

signaling a process manager to launch said intended recipient application.

4. The method of claim 1, wherein said launching said call director unit to set up said demon conference component includes:

loading a call processing module into said memory; and

initializing said call processing module to process calls using said network interface.

5. The method of claim 4, wherein said loading said call processing module into said memory comprises:

loading a call directing component;

loading a first conference component;

loading a first transport component; and

loading a first network component.

6. The method of claim 5, wherein said initializing said call processing module comprises:

initializing said first network component to operate with said network interface;

initializing said call directing component to monitor for said incoming call signal;

initializing said first transport component to receive said incoming call signal; and

initializing said first conference component to transfer said incoming call signal.

7. The method of claim 1, further comprising:

receiving an initialization message from said intended recipient application; and

removing said intended recipient application from an internal list if said initialization message does not correspond to an expected message.

8. In a computer system having a memory, a processor, and a network interface, an apparatus comprising:

a call directing module;

a process manager coupled to said call directing module; and,

a conferencing component coupled to said network interface and said call directing module;

where said conferencing component is configured by said call directory module to notify said call directing module upon receipt of an incoming call and causing said call director to signal said process manager to activate a conferencing application based on a listen string, said listen string containing an application signature, an application signal type, and an application signal port.

9. An apparatus comprising:

a processor;

a memory coupled to said processor;

a network interface coupled to said processor;

said memory configured to cause said processor to:

receiving an incoming call signal on said network interface;

processing said incoming call signal to detect an intended recipient application using a listen string, said listen string containing an application signature, an application signal type and an application signal port; and

launching a conferencing application using said application signature.

10. In a computer system having a memory, a processor, and a network interface, an apparatus comprising:

means for launching a call director unit to set up a demon conference component in said memory;

means for receiving an incoming call signal on said network interface;

means for processing said incoming call signal in said demon conference component to detect an intended recipient application using a listen string, said listen string containing an application signature, an application signal type and an application signal port; and

means for launching said intended recipient application using said application signature.

11. The apparatus of claim 10, wherein said means for processing said incoming call signal comprises:

means for parsing said incoming call signal to determine a signal type and a signal port; and

means for determining said intended recipient application based on said signal type and said signal port.

12. The apparatus of claim 10, wherein said means for launching said intended recipient application comprises:

means for determining said intended recipient application based on a signal type and a signal port;

means for locating said intended recipient application using said application signature; and

means for signaling a process manager to launch said intended recipient application.

13. The apparatus of claim 10, further comprising:

means for loading a call processing module into said memory; and

means for initializing said call processing module to process calls using said network interface.

14. The apparatus of claim 13, wherein said means for loading said call processing module into said memory comprises:

means for loading a call directing component;

means for loading a first conference component;

means for loading a first transport component; and

means for loading a first network component.

15. The apparatus of claim 14, wherein said means for initializing said call processing module comprises:

means for initializing said first network component to operate with said network interface;

means for initializing said call directing component to monitor for said incoming call signal;

means for initializing said first transport component to receive said incoming call signal; and

means for initializing said first conference component to transfer said incoming call signal.

16. The apparatus of claim 10, further comprising:

means for receiving an initialization message from said intended recipient application; and

means for removing said intended recipient application from an internal list if said initialization message does not correspond to an expected message.

17. An article comprising a computer readable medium having instructions stored thereon, which when executed, causes:

launching a call director unit to set up a demon conference component in a memory;

receiving an incoming call signal on a network interface;

processing said incoming call signal in said demon conference component to detect an intended recipient application using a listen string, said listen string

containing an application signature, an application signal type and an application signal port; and

launching said intended recipient application using said application signature.

18. The article of claim 17, wherein the computer readable medium further having instructions stored thereon, which when executed, causes:

parsing said incoming call signal to determine a signal type and a signal port; and

determining said intended recipient application based on said signal type and said signal port.

19. The article of claim 17, wherein the computer readable medium further having instructions stored thereon, which when executed, causes:

determining said intended recipient application based on a signal type and a signal port;

locating said intended recipient application using said application signature; and

signaling a process manager to launch said intended recipient application.

20. The article of claim 17, wherein the computer readable medium further having instructions stored thereon, which when executed, causes:

loading a call processing module into said memory; and

initializing said call processing module to process calls using said network interface.

21. The article of claim 20, wherein the computer readable medium further having instructions stored thereon, which when executed, causes:

loading a call directing component;

loading a first conference component;

loading a first transport component; and

loading a first network component.

22. The article of claim 21, wherein the computer readable medium further having instructions stored thereon, which when executed, causes:

initializing said first network component to operate with said network interface;

initializing said call directing component to monitor for said incoming call signal;

initializing said first transport component to receive said incoming call signal; and

initializing said first conference component to transfer said incoming call signal.

23. The article of claim 17, wherein the computer readable medium further having instructions stored thereon, which when executed, causes:

receiving an initialization message from said intended recipient application; and

removing said intended recipient application from an internal list if said initialization message does not correspond to an expected message.